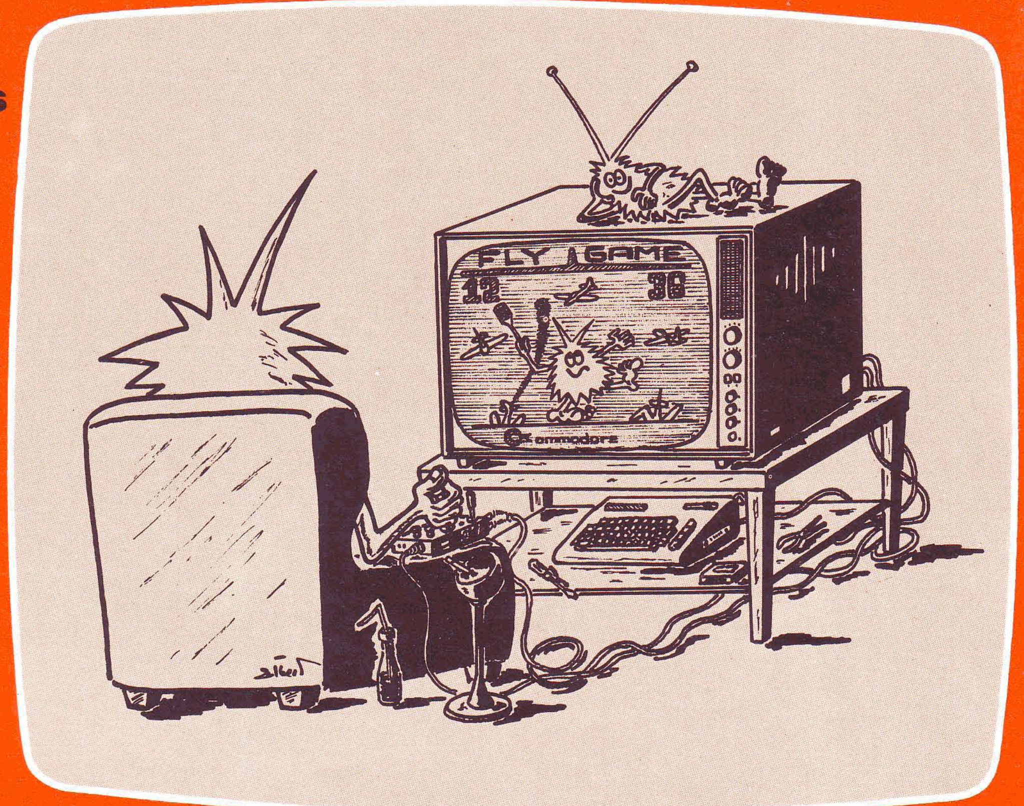
club commodore

Boletín informativo para los usuarios de microordenadores

VIC y CBM





julio 1983

TABLA DE EQUIVALENCIAS DEC-HEX-ASCII-PANTALLA-MNEMÓNICOS 6502 (VIC-20) I

DECIMAL	HEX	ASCII	PANTALLA	BASIC	6502	DECIMAL
0	00		@	end-line	BRK	0
1	01		A		ORA(I,X)	1
2	02		B			2
3	03		C			3
4	04		D			4
5	05		E		ORA Z	5
6	06		F		ASL Z	6
7	07		G			7
8	08		H		PHP	8
9	09		I		ORA #	9
10	0A		J		ASL A	10
11	0B		K			11
12	0C		L			12
13	0D	car ret	M		ORA	13
14	0E		N		ASL	14
15	0F		O			15
16	10		P		BPL	16
17	11	cur down	Q		ORA(I),Y	17
18	12	reverse	R			18
19	13	cur home	S			19
20	14	delete	T			20
21	15		U		ORA Z,X	21
22	16		V		ASL Z,X	22
23	17		W			23
24	18		X		CLC	24
25	19		Y		ORA Y	25
26	1A		Z			26
27	1B		[			27
28	1C		\			28
29	1D	cur right	]		ORA X	29
30	1E		↑		ASL X	30
31	1F		←			31
32	20	space	space	space	JSR	32
33	21	!	!	!	AND(I,X)	33
34	22	"	"	"		34
35	23	#	#	#		35
36	24	$	$	$	BIT Z	36
37	25	%	%	%	AND Z	37
38	26	&	&	&	ROL Z	38
39	27	'	'	'		39
40	28	(	(	(	PLP	40
41	29	)	)	)	AND #	41
42	2A	*	*	*	ROL A	42
43	2B	+	+	+		43
44	2C	,	,	,	BIT	44
45	2D	-	-	-	AND	45
46	2E	.	.	.	ROL	46
47	2F	/	/	/		47
48	30	Ø	Ø	Ø	BMI	48
49	31	1	1	1	AND(I),Y	49
50	32	2	2	2		50
51	33	3	3	3		51
52	34	4	4	4		52
53	35	5	5	5	AND Z,X	53
54	36	6	6	6	ROL Z,X	54
55	37	7	7	7		55
56	38	8	8	8	SEC	56
57	39	9	9	9	AND Y	57
58	3A	:	:	:	CLI	58
59	3B	;	;	;		59
60	3C					60
61	3D	=	=	=	AND X	61
62	3E				ROL X	62
63	3F	?	?	?		63

DECIMAL	HEX	ASCII	PANTALLA	BASIC	6502	DECIMAL
64	40				RTI	64
65	41	A	,a	A	EOR(I,X)	65
66	42	B	,b	B		66
67	43	C	,c	C		67
68	44	D	,d	D		68
69	45	E	,e	E	EOR Z	69
70	46	F	,f	F	LSR Z	70
71	47	G	,g	G		71
72	48	H	,h	H	PHA	72
73	49	I	,i	I	EOR #	73
74	4A	J	,j	J	LSR A	74
75	4B	K	,k	K		75
76	4C	L	,l	L	JMP	76
77	4D	M	,m	M	EOR	77
78	4E	N	,n	N	LSR	78
79	4F	O	,o	O		79
80	50	P	,p	P	BVC	80
81	51	Q	,q	Q	EOR(I),Y	81
82	52	R	,r	R		82
83	53	S	,s	S		83
84	54	T	t,	T		84
85	55	U	u,	U	EOR Z,X	85
86	56	V	,v	V	LSR Z,X	86
87	57	W	,w	W		87
88	58	X	,x	X	CLI	88
89	59	Y	,y	Y	EOR Y	89
90	5A	Z	,z	Z		90
91	5B					91
92	5C					92
93	5D				EOR X	93
94	5E		,		LSR X	94
95	5F		,			95
96	60				RTS	96
97	61				ADC(I,X)	97
98	62					98
99	63					99
100	64					100
101	65				ADC Z	101
102	66				RCR Z	102
103	67					103
104	68				PLA	104
105	69		,		ADC #	105
106	6A				RCR A	106
107	6B					107
108	6C				JMP(I)	108
109	6D				ADC	109
110	6E				ROR	110
111	6F					111
112	70				BVS	112
113	71				ADC(I),Y	113
114	72					114
115	73					115
116	74					116
117	75				ADC Z,X	117
118	76				ROR Z,X	118
119	77					119
120	78				SEI	120
121	79				ADC Y	121
122	7A		,			122
123	7B					123
124	7C					124
125	7D				ADC X	125
126	7E				ROR X	126
127	7F					127

EDITORIAL

ya llegan las vacaciones y M. E. C. SOFT. desembarca en "CLUB COMMODORE"

Como ya ha llegado el verano y —según sesudos estudios demuestran ampliamente— la actividad de programar produce un incremento de la temperatura corporal (sobre todo en la zona del craneo), vamos a dedicar este número a listar unos cuantos juegos. El aspecto térmico parece que no afecta a la gente de M.E.C. SOFT., entre otras razones porque tienen un estupendo acondicionador de aire en su cubil. En este ejemplar no sólo continúan con sus secciones habituales VENTANA CBM Y FICHEROS RELATIVOS sino que además inician una serie muy importante que esperamos que interese a muchos de nuestros lectores: ALMACENAMIENTO EN DISCOS que a cargo de M. AMADO tratará de explicar, de manera comprensible para cualquier persona interesada, el funcionamiento de TODAS las unidades de disco de COMMODORE y la manera de utilizarlas. También debemos mencionar —aunque no como novedad de este número— la serie de artículos de J. SASTRE sobre un tipo de ficheros muy útiles: los RELATIVOS. Para finalizar con el desembarco de M.E.C. SOFT. en CLUB COMMODORE, R. NAVARRO se ha hecho cargo del manejo de la VENTANA CBM que esperamos abra nuevos horizontes a usuarios y programadores. Después de todo esto, BUG parece seriamente desmoralizado.

¡Bien! Ahora a tomarse el verano de la manera más fresca posible y, como es tradicional en estos casos, a llenarse la mente de buenos propósitos para estas vacaciones (como preparar unas cuantas colaboraciones, por ejemplo).

Volveremos en septiembre. La próxima edición aparecerá después de los rigores estivales. ■

VENTANA CBM

modo de utilizar variables desde código máquina

por **RAFAEL NAVARRO** (M.E.C. SOFT.)

Releyendo el artículo aparecido en el número 8 de esta «Ventana CBM», he observado que, al final de la rutina de búsqueda de una variable numérica, se ejecuta un salto a lflp, rutina cuya dirección resultará desconocida para vosotros, ya que no se menciona en ninguna parte. Dicho JMP debe ser sustituido por JMP MOVFM. Esta subrutina (MOVFM) se encuentra (siempre en BASIC 4.0, series 4000 y 8000 de Commodore) en la dirección $CCD8. Por otro lado, los valores correctos que debe contener INTFL (flag entero-coma flotante) son $80 para enteros y $00 para coma flotante. La rutina de búsqueda de una variable numérica del BASIC (BVAR en $C187) coloca en A (acumulador) e Y (registro Y) los pesos bajo y alto, respectivamente, de la dirección de memoria donde se encuentra el valor de la variable buscada (2 bytes más adelante del nombre de la variable). La rutina MOVFM carga en el acumulador flotante # 1 (FAC) 5 bytes a partir del punto de memoria indexado por el acumulador y el registro Y. Así pues, llamando a MOVFM después de BVAR, dispondréis del valor de la variable en el acumulador flotante # 1. Siempre se emplean cinco bytes, independientemente de que la variable sea entera

(continúa en la pág. siguiente)

VENTANA CBM

modo de utilizar variables desde código máquina

(viene de la pág. anterior)

o de coma flotante, aunque la representación interna sea diferente (solamente dos bytes) si la variable es entera.

Aunque la precisión de almacenamiento en variables de coma flotante es de cinco bytes, los acumuladores tienen una longitud de seis bytes para lograr una mayor precisión en las operaciones matemáticas internas que siempre se realizan entre los acumuladores. En próximos capítulos, comentaremos algunas de las rutinas matemáticas del sistema operativo. Gracias a su utilización, vereis cómo puede ganarse mucho tiempo de ejecución operando directamente en código máquina con el apoyo de las rutinas del sistema operativo y del interpretador. Veamos ahora cómo asignar un valor a una variable determinada. Para variables numéricas, consideraremos el caso de asignar un valor dado a la variable entera AB %. Para ello supondremos que el valor a asignar está disponible en el acumulador # 1 (alguna rutina anterior lo dejó allí).

Utilizaremos aquí la rutina de sistema operativo MOVMF (SCD0A) que transfiere el contenido del FAC al punto de memoria indexado por X (registro X) e Y en vez de A e Y, por lo cual será necesario realizar un TAX —ver subrutina ejemplo— antes de llamarla. ■

SUBRUTINA DE ASIGNACIÓN DEL CONTENIDO DE FAC A UNA VARIABLE NUMÉRICA

```
ASVNUM                       ; ejemplo de asignación numérica con
                               FAC
            JSR BUVN         ; búsqueda variable para cargar A e Y
                               con la dirección de AB%
            TAX              ; carga del registro X con el valor del
                               acumulador, ya que la rutina MOVMF
                               requiere el puntero en X e Y en vez
                               de en A e Y (tal y como lo entrega
                               BUVN)
            JSR MOVMF        ; carga de AB% con el contenido de
                               FAC
            RTS

BUVN        LDA # 'A
            LDX # 'B
            STA VARNAM
            STX VARNAM+1
            LDA # $00
            STA VALTYP       ; variable numérica
            LDA # $80        ; $80 ó decimal 128 = entero
            STA INTFL
            JSR BVAR
            RTS
```

En el número anterior hablamos de los Ficheros Relativos en BASIC 4. Más de un propietario de «VIC-20» o «Commodore 64» habrá quedado decepcionado, pues estos aparatos están dotados de BASIC 2 y, por tanto, no disponen de las instrucciones DOPEN #, DCLOSE # y RECORD # para el manejo de los Relativos. Para solventarlo, este mes dividiremos el artículo en dos partes: en la primera explicaremos cómo trabajar con los Relativos utilizando el BASIC 2 y, en la segunda, describiremos de manera general el tratamiento que el DOS da a los Relativos, cosa que afecta tanto al BASIC 2 como al BASIC 4.

FICHEROS RELATIVOS EN BASIC 2

En primer lugar, conviene saber que los Ficheros Relativos son una función del DOS (sistema operativo del disco), y no del OS (sistema operativo de la CPU). Pueden trabajar con Ficheros Relativos todas las unidades de discos dotadas de DOS 2.0 y ulteriores, que son los modelos siguientes:

Modelos	DOS
1540,1541,4040	2.1
8050	2.5
2031	2.6
8250	2.7
D9060, D9090	3.0

Existe una manera de enviar al disco las instrucciones DOPEN #, DCLOSE # y RECORD # utilizando el BASIC 2. Los ejemplos de la tabla 1, sacados del número anterior de esta Revista (ver en «Club Commodore» 9 el artículo de los Relativos en BASIC 4), muestran las equivalencias entre los dos BASICs.

Las instrucciones INPUT # y PRINT # son las mismas para los dos BASICs.

Para entender mejor los ejemplos, explicaremos el formato de estos nuevos «comandos» que hemos añadido al BASIC 2:

La instrucción OPEN es la misma que para los ficheros secuenciales, cambiando los últimos parámetros, por lo que queda de la siguiente manera:

OPEN nf,pe,ds,"dr:nombre,L"+CHR$(lr)

donde:

nf = Número de fichero lógico
pe = Número de periférico
ds = Dirección secundaria
dr = Número de drive
nombre = Nombre del fichero
lr = Longitud de los registros

FICHEROS

ficheros relativos en BASIC 2 (II)

por JORDI SASTRE

La instrucción CLOSE no tiene dificultad:

CLOSE nf

La RECORD # precisa de más explicación. Necesita tener abierto el canal de errores y comandos (#15), por ejemplo OPEN 15,8,15, y su formato es el siguiente:

PRINT #15,"P"+CHR$(96+ds)+ CHR$(rb)+CHR$(ra)+CHR$(1)
donde:

ds = Dirección secundaria del fichero (ver OPEN).
rb = peso bajo del número de registro.
ra = Peso alto del número de registro.

Aquí vemos que el número de registro debe entregarse empaquetado en dos bytes. La manera de hacerlo es muy simple: siendo NR el Número de registro:

Peso bajo = INT(NR-INT(NR/256)*256)
Peso alto = INT(NR/256)

En adelante, cuando hablemos de Ficheros Relativos, los ejemplos estarán escritos en BASIC 4, por tener las instrucciones propias de los Ficheros Relativos pero, con lo dicho en este artículo, no tendrá ningún problema el que quiera trabajar con BASIC 2.

TABLA 1

BASIC 4	BASIC 2
DOPEN #1,"FICHERO",L50	OPEN1,8,4,"0:FICHERO,L,"+ CHR$(50)
DOPEN #1,"FICHERO"	OPEN1,8,4,"0:FICHERO"
DCLOSE #1	CLOSE1
RECORD #1,100	PRINT #15,"P"+CHR$(96+4)+ CHR$(100)+CHR$(0)+CHR$(1)
RECORD #1,(NR)	PRINT #15,"P"+CHR$(96+4)+ CHR$(NR-INT(NR/256)*256)+ CHR$(INT(NR/256))+CHR$(1)

DESCRIPCIÓN GENERAL DE LOS FICHEROS RELATIVOS

Los Ficheros Relativos se llaman así porque permiten al programador posicionarse en cualquier punto del fichero indicando su posición relativa con respecto al principio del mismo. En el directorio del disco se identifican mediante «REL». Comparados con los Ficheros Secuenciales, existe esencialmente la siguiente diferencia: el Fichero Secuencial forma un único conjunto con la información que contiene y para actualizar un dato debe leerse todo el fichero, efectuar la actualización en la memoria de la CPU y después volver a grabar de nuevo todo el fichero en el disco. En cambio, el Fichero Relativo se divide en diferentes zonas, que llamaremos «registros». Cada registro contiene un solo registro lógico de manera que, cuando se desean actualizar datos, basta con gestionar los registros pertinentes, permaneciendo inalterable el contenido restante del fichero.

Un Fichero Relativo se compone de tres partes:

— Los «bloques» de datos del usuario.
— Los «sector-índices», de uso interno del DOS, que direccionan sobre los bloques de datos.
— Un «super sector-índice», también de uso interno del DOS y sólo en los modelos 8250, D9060 y D9090, que direcciona sobre los «sector-índices».

Esta estructura, semejante a una estructura de árbol, es ajena totalmente al trabajo del programador con los ficheros. No obstante, vamos a explicarla más a fondo:

Un fichero se compone de registros. Cada registro es la zona donde el programador almacena los datos correspondientes a un registro lógico. Es el mismo programador quien define el número y la longitud de los registros. La longitud máxima de un registro es de 254 bytes. Más adelante hablaremos del número máximo de registros. El programador accede al fichero por medio de los registros, es decir, lee y graba datos en el fichero indicando simplemente el número de registro con el que desea trabajar.

Internamente, el DOS agrupa los registros en bloques de 256 bytes cada uno. Si en un bloque no cabe un número exacto de registros, el DOS reparte el contenido de un registro entre dos bloques. El programador maneja los registros sin preocuparse para nada de los bloques. De la tarea de averiguar en qué bloque o bloques está contenido un registro se encarga el DOS. Para ello dispone de los sector-índices. Un ejemplo de la estructura registro-bloque puede ser el de la figura 1.

Como vemos, el registro 1 cabe perfectamente dentro del bloque 1; pero el registro 2 está repartido entre

(continúa en la pág. siguiente)

ficheros relativos en BASIC 2

(viene de la pág. anterior)

los bloques 1 y 2; el registro 3 empieza en el bloque 2 pero tiene sus últimos bytes en el 3; el registro 4 está íntegro en el bloque 3; etc.

Cuando los registros tienen una longitud de 1, 2, 127 ó 254 bytes (submúltiplos de 254), en un bloque cabe un número exacto de registros y no es necesaria ninguna partición. Pero, por desgracia, éste es el caso menos frecuente.

Cuando desea leerse un registro que está repartido entre dos bloques, el DOS carga en buffer todo el contenido de los dos bloques de datos en que reside el registro, es decir, carga el registro junto con sus inmediatos vecinos. Esta manera de trabajar es muy ventajosa en cuanto a velocidad. Las lecturas y escrituras de los registros existentes en buffer se realizan sobre el mismo buffer y no son actualizadas en el disco hasta que el programador se posiciona sobre un registro que no está contenido en la zona de datos actualmente en buffer. Entonces, siempre que en este buffer de disco hubiera dos o más registros, el programador podrá trabajar con ellos sin que en ningún momento sea necesario utilizar el disco. Sólo cuando el fichero se cierre, o se acceda a uno de los registros que en aquel momento no estén en buffer, el DOS se verá obligado a grabar el contenido actualizado de estos registros en el disco.

Un sector-índice es un bloque de disco (256 bytes) que apunta hasta 120 bloques de datos. Por medio de ellos, el DOS puede calcular, según la longitud y el número de registros del fichero, en qué bloques está contenida la información correspondiente a un registro determinado. El Fichero Relativo puede disponer de un máximo de seis sector-índices, que direccionan sobre 720 (6 × 120) bloques de datos, o sea sobre 182.880 (720 × 254) bytes útiles para el programador, sea cual sea el número de registros y la longitud de los mismos.

El formato de un sector-índice es el siguiente:

Byte	Contenido
0-1	Dirección (Pista-Sector) al próximo sector-índice.
2	Número de sector-índice en modelos 1540, 1541, 4040, 8050. $FE constante en modelos 8250, D9060, D9090.
3	Longitud del registro.
4-5	Dirección (Pista-Sector) del primer sector-índice.
6-7	Dirección (Pista-Sector) del segundo sector-índice.
8-9	Dirección (Pista-Sector) del tercer sector-índice.
10-11	Dirección (Pista-Sector) del cuarto sector-índice.
12-13	Dirección (Pista-Sector) del quinto sector-índice.
14-15	Dirección (Pista-Sector) del sexto sector-índice.
16-17	Dirección (Pista-Sector) del bloque de datos 1.
18-19	Dirección (Pista-Sector) del bloque de datos 2.
(...)	(...)
254-255	Dirección (Pista-Sector) del bloque de datos 120.

Para calcular la ocupación de un Fichero Relativo en el disco debe efectuarse lo siguiente: primero, multiplicar el número de registros por su longitud para obtener el número de bytes que se necesitan; seguidamente, dividir los bytes por 254, redondeando hasta el próximo entero, para saber los bloques de datos necesarios para albergar los bytes. Dividiendo el número de bloques entre 120 (redondeando de nuevo al próximo entero), sabremos los sector-índices necesarios que, sumados al número de bloques de la anterior operación, nos dará el número total de bloques de disco que ocupará el Fichero Relativo.

He mencionado antes que los Ficheros Relativos de los modelos 8250, D9060 y D9090 disponen de un «super sector-índice». Ello no es más que un sector-índice que direcciona sobre un máximo de 127 sector-índices del mismo modo que un sector-índice direcciona sobre los bloques de datos. Así se consigue (sólo en los modelos mencionados) multiplicar por 127 la capacidad máxima del Fichero Relativo (23.225.760 bytes), a costa de un bloque más de ocupación del mismo en el disco. Pero, ¡atención!: a esta capacidad no puede llegarse nunca, pues existe un tope de 65.535 registros por fichero. Multiplicando 65.535 (número máximo de registro) por 254 (número máximo de bytes por registro) nos da la capacidad máxima del Fichero Relativo en los modelos mencionados, que es de 16.645.890 bytes ó 66.086 bloques. Pero de nuevo nos hallamos ante una nueva cifra teórica, pues este volumen supera la capacidad de cualquiera de estos tres modelos. Por tanto, baste decir que los Ficheros Relativos en los modelos 8250, D9060 y D9090 no tienen límite de capacidad.

El Fichero Relativo utiliza tres buffers de disco (uno para el sector-índice y dos para bloques de datos), a diferencia del Fichero Secuencial que usa sólo dos. Puesto que existe un total de diez buffers disponibles para ficheros, únicamente pueden mantenerse abiertos a la vez tres Ficheros Relativos.

Cada registro sólo puede grabarse mediante una única instrucción PRINT #, al término de la cual el DOS rellena las restantes posiciones del registro (si quedaran) con ceros binarios. Esto también implica que cuando el DOS lee un registro, lo hace hasta el último byte que no es cero, tras lo cual se posiciona automáticamente en el inicio del siguiente registro.

¡Atención al párrafo anterior! Si en un registro se tienen, por ejemplo, 50 bytes de información, no pueden actualizarse únicamente los 10 primeros pensando que los restantes 40 van a permanecer intactos: deben re-grabarse los 50 de nuevo. Y, por otra parte, si el programador desea grabar información empaquetada en un registro, debe usar, como terminador de registro, un carácter de retorno de carro (CHR$(13)) para salvaguardar los ceros binarios que formen parte del empaquetado.

En el próximo número... ¡hablaremos de Relativos!

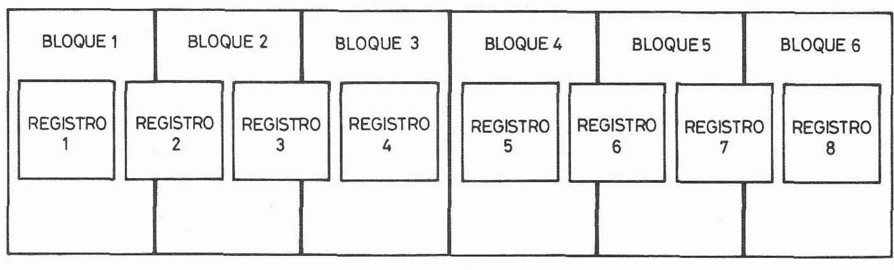


Fig. 1

SOFTWARE DE BASE (V)



funciones CHR$ y ASCII

por E. MARTÍNEZ DE CARVAJAL

Estas dos funciones ya están presentes en el BASIC del VIC-20 pero su funcionamiento difiere del de algunas otras versiones de BASIC. En el VIC-20 estas funciones sirven para obtener el código ASCII de un carácter o para obtener éste a partir de su valor ASCII. En ambos casos, sólo se maneja un carácter o, lo que es lo mismo, un literal de longitud 1, siendo su mayor utilidad el control de entradas de datos mediante la sentencia GET. Sin embargo, es fácil observar que, puesto que hay 255 códigos ASCII, en un solo carácter se pueden guardar y recuperar, mediante estas dos funciones, números enteros positivos entre 0 y 255, lo cual supone un gran ahorro de memoria respecto de las funciones usuales STR$ y VAL.

```
A=123
A$=STR$ (A)
PRINT LEN(A$)
4
PRINT VAL(A$)
123
A$=CHR$(A$)
PRINT LEN(A$)
1
PRINT ASC(A$)
123
```

Como podéis ver, para guardar el mismo valor, en un caso se utilizan cuatro caracteres y en el otro sólo uno. Algunas versiones de BASIC han buscado la máxima utilidad de estas funciones precisamente en esta faceta, permitiendo aplicarlas a literales de más de un carácter de longitud, lo cual supone un considerable ahorro de espacio al manejar cantidades numéricas convertidas a literales. En nuestro caso, para no complicar excesivamente las rutinas, hemos de desperdiciar algo de espacio para prever los casos de números negativos y/o con hasta dos decimales. Aún así, y como podéis ver en la siguiente tabla, el ahorro de memoria es considerable.

Núm. caracteres	Valores que puede contener
3	± 655,32
4	± 167772,07
5	± 4294967,30
n	± [2^(n-1)*81/100

Las rutinas se dan en los listados adjuntos. ■

```
PROGRAMA:10SOFTBASECHR$

READY.

9000 REM CHR$
9001 REM
9002 REM DATOS DE ENTRADA
9003 REM
9004 REM X1=NUM. A COMPACTAR
9005 REM X2=EN CUANTOS CARACTERES
9006 REM
9007 REM DATOS DE SALIDA
9008 REM
9009 REM X$=NUMERO COMPACTADO
9010 X$="":FORI=1TOX2:X$=X$+CHR$(0)
:NEXTI
9015 I=X2
9020 N1=ABS(X1)*100
9030 N2=INT(N1/256)
9040 N3=N1-(N2*256)
9050 N1=N2
9060 X$=LEFT$(X$,I-1)+CHR$(N3)+RIGH
T$(X$,X2-I)
9067 I=I-1
9070 IF N1=0 OR I=1 THEN 9090
9080 GOTO 9030
9090 IF X1<0 THEN X$="-"+RIGHT$(X$,
X1-1)
9100 RETURN
READY.
```

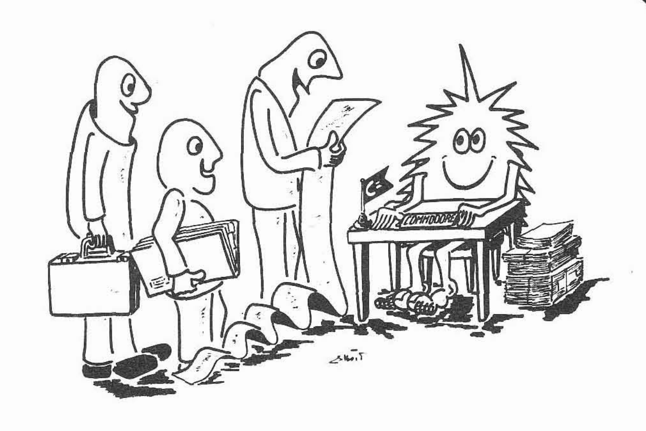
```
PROGRAMA:10SOFTBBASEASCII

READY.

9200 REM ASCII
9205 X1=0
9210 FORI=X2-1TO2STEP-1
9220 X1=(X1+ASC(MID$(X$,X2-I+1,1)))
*256
9230 NEXTI
9240 X1=X1+ASC(MID$(X$,X2,1))
9250 IFLEFT$(X$,1)="-"THENX1=-X1
9260 X1=X1/100
9270 RETURN
READY.
```

COLABORACIONES

"topo loco"



por **MANUEL TORRALBA**

¡Hola!

Soy un vic-cioso de Barcelona. Recibo «Club Commodore» en mi casa cada mes y, como pedís colaboraciones, os mando este programa de juegos.

Mi nombre es Manuel Torralba y *tengo 13 años* y el juego, que va en la cassette que os adjunto, se llama «Topo Loco». No necesita ninguna ampliación de memoria. El jugador debe «limpiar» el huerto (la pantalla) de lechugas (representadas por tréboles) sin chocar con los bloques grises que hay por todo el huerto, ni tampoco con el rastro que deja. Cuando acabas un huerto pasas a otro, y *así su-*

(termina en la pág. siguiente)

LISTADO

```
PROGRAMA:10TOPO1

0 REM****************
1 REM*  TOPO LOCO   *
2 REM*--------------*
3 REM*   MANUEL     *
4 REM*              *
5 REM*  TORRALBA    *
6 REM****************
7 MN=1:GUS=3:GOSUB3000
8 POKE36879,8:FORT=1TO5:FORR=1TO210
:NEXTR:PRINT"[ CLR ]"
9 FORR=1TO210:NEXT:PRINT"[HOME ][CRSRD]
[CRSRD][CRSRD][CRSRD][CRSRD][CRSRD]
[CRSRD][CRSRD][CRSRD][CRSRD][CRSRD]
[CRSRR][CRSRR][CRSRR][CRSRR]HUERTO"
;MN:NEXTT
10 LECH=15:POKE36879,8:PRINT"[ CLR ]
":POKE36878,15
11 POKE36879,8:PRINT"[ CLR ]":FORL=
1TO15+P1:POKE7680+INT(RND(1)*506),1
02:NEXTL
12 FORL=1TOLECH:POKE7680+INT(RND(1)
*506),88
13 POKE36878,15:POKE36876,0:FORE=1T
O120:NEXTE:POKE36876,235+L:NEXTL
14 FORE=1TO200:NEXTE
20 B=D:M=0:X=10:Y=10:C=255
25 GETTECLA$
30 IFTECLA$=""THENGOTO80
40 IFTECLA$=" "THENM=1
50 IFTECLA$="I"THENM=2
60 IFTECLA$="J"THENM=0
70 IFTECLA$="L"THENM=3
80 IFM=0THENX=X-1
81 POKE36878,15:POKE36876,190
90 IFM=1THENY=Y+1
100 IFM=2THENY=Y-1
110 IFM=3THENX=X+1
120 IFPEEK(7680+22*Y+X)<>88ANDPEEK(
7680+22*Y+X)<>32THEN GOTO150
121 IFPEEK(7680+22*Y+X)<>88THEN GOT
O130
122 POKE36878,15:FORN=150TO250:POKE
36876,N:NEXTN:POKE36874,0:SC=SC+20:
LECH=LECH-1
123 IFLECH=0THENGOTO9000
130 POKE7680+22*Y+X,81
135 FORA=1TOB:NEXTA:B=B-0.2
140 POKE36876,0:GOTO25
150 POKE7680+22*Y+X,42
155 POKE36878,15
160 POKE36876,0
161 FOR A=15TO0STEP-0.081:C=C-1:POK
E36878,A:POKE36879,C:POKE36877,180:
NEXTA:POKE36877,0
162 GUS=GUS-1
163 IFGUS=0THEN9019
168 GOTO11
190 POKE36878,15
1015 FORR=1TO100:NEXT
3000 PRINT"[ CLR ]":POKE36879,8:POK
E36865,140
3010 PRINT"[CRSRD][CRSRD][ WHT ][CRSRR]
[CRSRR]**TOPO LOCO**  ":PRINT"[CRSRD]
[CRSRD][CRSRR]MUEVE AL TOPO LOCO "
3011 PRINT"[CRSRD][CRSRR]Y COME TOD
AS":PRINT"[CRSRD][CRSRR]LAS LECHUGA
S([SHIFX])   "
3012 PRINT"[CRSRD][CRSRR]EVITANDO C
HOCAR"
3020 PRINT"[CRSRD]CON TU RASTRO O L
OS[CRSRR][CRSRR][CRSRR][CRSRD]BLOQU
ES GRISES"
3021 PRINT"[CRSRD][CRSRR]DIRIGE EL
TOPO CON"
3022 PRINT"[CRSRD][CRSRR] I,J,L,[RVSON]
SPACE[RVSOF]"
3030 PRINT"[CRSRD][CRSRR]  POR M.TO
RRALBA "
3060 FORALTURA=140 TO 40STEP-0.11:P
OKE36865,ALTURA:NEXTALTURA
3090 FORR1=1TO3000:NEXTR1:POKE198,0
3091 PRINT"[ CLR ][CRSRD][CRSRD][CRSRD]
[CRSRD][CRSRD][CRSRD][CRSRD](1 ES E
L MAS DIFICIL)"
4000 INPUT"[HOME ][ WHT ][CRSRD][CRSRD]
[CRSRD][CRSRD]DIFICULTAD (1-10)";D
4010 D=D*15 :RETURN
9000 POKE36878,15
9011 FORR=1TO34 :READT:POKE36876,T:
POKE36879,T:FORM=1TO150:NEXTM:NEXTR
:RESTORE
9012 P1=P1+5:MN=MN+1:GOTO8
9019 POKE36878,15
9020 FORU=1TO34:READA:NEXTU
9021 POKE36879,8:FORR=1TO19 :READT:
POKE36876,T:FORM=1TO150:NEXTM:NEXTR
9022 PRINT"[HOME ][CRSRR][CRSRR][CRSRR]
[CRSRR][CRSRR][CRSRR][CRSRR][CRSRD]
[CRSRD][CRSRD][CRSRD][CRSRD][CRSRD]
[CRSRD]GAME OVER":FORR=1TO3000:NEXT
R:RUN
20000 DATA0,0,8,195,209,209,215,219
,209,219,215
20010 DATA195,209,209,215,219,209,2
09,207,209,209,215,219,221,219,215
20020 DATA209,207,195,201,207,209,2
09,8,8
20030 DATA0,0,225,223,0,219,215,0,2
09,207,201,207,201,195,0,0,225,0

READY.
```

TRUCOS

Pokes variados para el VIC-20 y el COMMODORE 64

por **P. MASATS**

Se plantean normalmente algunas dudas relacionadas con el funcionamiento de los periféricos del VIC-20 en el COMMODORE 64. Vamos a intentar aclararlas. Tanto la impresora VIC 1515 como la unidad de cassette funcionan directamente en el COMMODORE 64. No obstante, utilizando el cassette ocurre lo siguiente: después de dar la orden LOAD «LO QUE SEA» aparece el clásico mensaje PRESS PLAY ON TAPE. En cuanto pulsamos la tecla PLAY la información de la pantalla desaparece y la cinta comienza a moverse. Al encontrar un programa o un fichero, la cinta se para y reaparece la información visual indicando FOUND «LO QUE SE HAYA ENCONTRADO» y espera unos segundos antes de continuar. Si deseamos prescindir de esta espera podemos pulsar la tecla «COMMODORE» (la del logotipo) y así inmediatamente continúa la lectura de la cinta (apagándose la pantalla) hasta que ha finalizado la operación. Por lo que respecta al disco ocurre lo siguiente: En estos momentos se comercializa por Microelectrónica y Control, S. A., una única unidad de disco VIC 1541 que es compatible con el VIC y el COMMODORE 64 sin ningún tipo de problema. Para aquellos que, por una razón u otra, posean una unidad de disco modelo VIC 1540 existe una ROM que permite convertirla en la adecuada para el COMMODORE 64. No obstante, mediante unos POKEs es posible utilizar el modelo 1540:

Al ir a cargar o grabar un programa se debe teclear: POKE53265,11:LOAD «NOMBRE DEL PROGRAMA»,8 (RETURN). En caso de grabar hacer lo mismo pero sustituyendo el LOAD por SAVE o VERIFY.

La operación anterior hace desaparecer el contenido de la pantalla como ocurre con la operación del cassette, excepto que no se recupera automáticamente. Para ello debe hacerse POKE 53265,27 o pulsar simultáneamente las teclas RUN/STOP y RESTORE. ■

"topo loco"

(viene de la pág. anterior)

cesivamente hasta agotar los tres topos que posees. Para hacerlo más difícil, la velocidad del topo se incrementa poco a poco y, por cada huerto, el número de bloques aumenta en cinco. Y además, aunque me parecía difícil, le añadí música, color y sonido.

A continuación os explico el programa:

Línea 1:
Define el número de topos y huertos y continúa en la línea 3000.

Línea 8 a 9:
Ponen en pantalla el número de huertos.

Línea 10:
Define el número de lechugas.

Línea 11 y 12:
Colocan las lechugas y los bloques en su lugar.

Línea 20:
Define velocidad de juego, posición XY y la variable de color.

Línea 25 a 110:
Movimiento del topo.

Línea 120 y 121:
Comprueban qué objeto tiene delante el topo. Si es un bloque, pasa a la línea 150.

Línea 122:
El topo ha encontrado una lechuga.

Línea 123:
Caso de 0 lechugas, pasa a la línea 9000.

Línea 130:
Pone al topo en su posición.

Línea 135:
Retardo (velocidad).

Línea 140:
Vuelve a la línea 25.

Línea 150 a 168:
Explosión y un topo menos. Caso de 0 topos, pasan a la línea 9019.

Línea 3000 a 3090:
Instrucciones.

Línea 4000 a 4010:
Piden nivel dificultad.

Línea 9000 a 9012:
Música éxito (huerto vacío).

Línea 9019 a 9021:
Música de fin de juego.

Línea 9022:
Mensaje fin de juego.

Línea 20000 a 20030:
Datas musicales.

Principales variables: número de huertos (mn), número de topos (gus), número de lechugas (lech), velocidad (b), bloques adicionales (pl), posición del topo (X e Y).

Esperando noticias vuestras, se despide un colaborador **Manel**

P.D. La carta la ha pasado a máquina mi hermano mayor, que ha corregido todas (o casi todas) las faltas y ha puesto el rollo chulo. EL PROGRAMA ES MÍO, LO HE HECHO YO SOLITO Y NADIE ME HA AYUDADO. (¡Es que nadie se cree que yo pueda programar así!) ■

BANCO DE PRUEBAS

los tres nuevos programas para manejo de ficheros en cinta de Microelectrónica y Control

por **P. MASATS**

Le toca el turno en el banco de pruebas a un conjunto de programas que promete ser de gran utilidad para los usuarios del VIC-20. Se trata de tres programas que, convenientemente adaptados, permiten la CREACIÓN, CONSULTA, MODIFICACIÓN, LISTADO y ORDENACIÓN de cualquier tipo de fichero en cinta.

Las modificaciones que ha de hacer el usuario para adaptar estos programas a sus necesidades se limitan a unas sentencias DATA y algunas variables claramente especificadas en los manuales. Los programas están totalmente escritos en BASIC, se pueden listar, modificar y guardar cuantas veces se desee para tener diversas versiones que permitan manejar diferentes ficheros. Los programas vienen preparados para trabajar con un fichero de ARTÍCULOS.

El primer programa, llamado MANTENIMIENTO DE FICHEROS (C-220), es el que permite crear, consultar y modificar el fichero.

El segundo programa, llamado LISTADO DE FICHEROS (C-221), es el que permite sacar por impresora los datos del fichero. Además de los datos relativos a los campos, hay que especificarle el formato de impresión mediante la modificación del programa original.

El tercer programa, cuya denominación es ORDENACIÓN DE FICHEROS (C-222), permite ordenar cualquier fichero por varios conceptos simultáneamente. Utiliza una rutina de alta velocidad que permite ordenar 100 registros en 19 segundos. El número máximo de registros a ordenar es de 999.

En este programa, los datos a especificar son los relativos a los conceptos por los cuales se desea la ordenación. Por ejemplo, podría ser por el nombre. Se puede asignar un nombre al fichero ordenado diferente del original y consultarlo, modificarlo y listarlo con versiones apropiadas de los programas anteriores.

Los tres programas van provistos de su correspondiente manual y los puede adquirir por separado al precio de 3.000 ptas. cada uno, en su distribuidor habitual. ■

MEA CULPA

CORRECCIÓN DE UN ERROR EN EL PROGRAMA «LEY DE OHM», DEL MES DE MAYO.

Existe un error en la línea 940 de dicho programa. Ésta debe ser:

940 IF G$ = "S" THEN PRINT : PRINT "V = "; P/I (el resto de la línea es el mismo).

PROGRAMAS 

PROGRAMAS EN DISCO

D-1001 Agenda 5.000
Permite contener 4 páginas de información de 114 personas, entidades, etc. Pudiendo añadir, cambiar o borrar la información que se desee.
En castellano, necesita ampliación de 8K. Presentado en caja.

D-1002 QSL 3.000
Registro de contactos para radioaficionados, versión en disco.
Necesita ampliación de 8K. Presentado en caja.

D-1003 Test Demo 3.000
Programa de test para la unidad de disco. Nota: este disco es el que se suministra con la unidad de disco VIC-1540.

D-1004 Assembler 5.000
Editor y compilador en BASIC para rutinas en lenguaje máquina del 6502. Necesita ampliación de 3K. Acompañado de manual en inglés.

D-1005 English language 2.500
Versión en disco del programa en cinta, dado que al manejar ficheros resulta mucho más rápida su utilización.
Necesita ampliación de 8K. Presentado en caja.

D-1006 Quiz Master 2.500
Versión en disco del programa en cinta. Necesita ampliación de 8K. Presentado en caja.

D-1007 Matemáticas 1 (nivel BUP) 2.500
Versión en disco del programa en cinta. Necesita ampliación de 8K. Presentado en caja.

3301 Simplicalc 13.000
Hoja de Trabajo Electrónica 60 columnas por 100 filas Max. Necesita ampliación de 16K.

3304 Vic File 13.000
Bases de datos para el VIC-20 con posibilidad de definir el formato de los campos. Necesita ampliación de 16K. Ideal para fichero de clientes.

3305 Vic Writer 13.000
Proceso de texto muy potente y rápido. Necesita ampliación de 8K ó 16 K.

PROGRAMAS EN CINTA

Programas Técnicos

C-128 Programación lineal 1.000
Método simplex. Cálculo del valor de las variables que satisfaciendo las restricciones hacen máxima o mínima una función. En castellano. Presentado en caja.

C-129 Matrices 1.000
Suma, resta, multiplicación, multiplicación por un escalar e inversión de matrices. En castellano. Presentado en caja.

C-131 Regresiones I 1.000
Contenido: Regresión lineal: Cálculo por mínimos cuadrados de la recta que se ajusta mejor a una nube de puntos.
Regresión múltiple: Variable dependiente en función de N variables independientes de grado 1. En castellano. Presentado en caja.

C-132 Regresiones II 1.000
Contenido: Regresión de orden N: Variable dependiente en función de variable independiente de grado N.
Regresión exponencial: Ajuste nube de puntos a una curva exponencial.
Regresión geométrica: Ajuste de una nube de puntos a una curva geométrica. En castellano. Presentado en caja.

C-133 Estadística I 1.000
Distribución normal, Poisson, binomial, chi-cuadrado, Student, F de Snedecor. En castellano. Presentado en caja.

C-134 Estadística II 1.000
Contenido: Cálculo de la media, varianza y desviación tipo, tanto de la muestra como de la población, estando los datos agrupados o no. Test de chi-cuadrado y test de Student. En castellano. Presentado en caja.

C-135 Sistemas 1.000
Contenido: Resolución de sistemas de N ecuaciones con N incógnitas.
Resolución de ecuaciones de grado 2 dando las soluciones tanto reales como complejas.
Cálculo de permutaciones y de combinaciones. En castellano. Presentado en caja.

C-137 Integración 1.000
Cálculo de integrales por el método de Gauss.
Cálculo de derivadas de una función en un punto. Interpolaciones tanto lineales como curvilíneas. En castellano. Presentado en caja.

Programas Educativos

C-125 Hangmath 1.500
Se trata de averiguar la multiplicación entre 2 números (tanto los números como la multiplicación y el resultado final), procurando no ser ahorcado. Las instrucciones son en castellano.

C-140 Skymath 1.500
Sumas y restas para chicos muy jóvenes saliendo naves espaciales para motivarlos. Una manera de aprender jugando. 3K.

C-141 Space Division 1.500
20 preguntas sobre divisiones muy sencillas. En función de los resultados correctos, despega un cohete que llegará más o menos lejos. 3K.

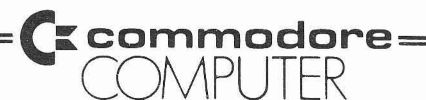
C-143 English Language 2.000
Especialmente adecuado para las personas que, poseedoras de un buen nivel de inglés, quieren alcanzar cotas superiores. Necesita ampliación de 8K. Presentado en caja y acompañado de su correspondiente manual.

C-145 Mastermind 3.000
Participe en el famoso concurso de la BBC, de preguntas y respuestas sobre temas variados.
Este cassette va acompañado del sistema operativo y de un file de demostración. Necesita ampliación de 8K. Presentado en caja.

C-146 Matemáticas I (nivel BUP) 2.000
Contiene sistema operativo y 7 cuestionarios sobre diferentes temas abarcando geometría, aritmética, matemáticas generales, álgebra, etc. Necesita ampliación de 8K ó 16K. Presentado en caja.

PROGRAMAS COMERCIALIZADOS POR "MICROELECTRÓNICA Y CONTROL, S.A."
Puede encontrarlos en su distribuidor más próximo

microelectrónica y control s a 





Este es el nuevo ordenador personal COMMODORE 64. Un gigante de 40 cm, con un precio casi tan pequeño como su tamaño.

Nadie hasta ahora había logrado ofrecerle 64 K de memoria, 40 columnas en pantalla, 8 sprites y un sonido de auténtica maravilla por sólo 110.000,— ptas. Claro que tampoco todo el mundo es el líder mundial en microordenadores.

COMMODORE sabe perfectamente que para seguir siendo el número uno, tiene que estar constantemente en vanguardia. De calidad. De precios. De todo. Para ello investigamos constantemente.

Afortunadamente nuestra labor se ve plenamente recompensada cuando vemos, como lo demuestra el cuadro comparativo, que nuestro más directo competidor cuesta nada menos que un 100% más caro. Y ello sin reunir todos los adelantos técnicos del COMMODORE 64.

---

1. Capacidad total de memoria RAM de 64 K. Interpretador BASIC extendido y sistema operativo residentes en ROM.

2. Dotado del más potente chip sintetizador de sonido diseñado hasta hoy, el COMMODORE 64 ofrece 3 voces totalmente independientes con una gama de 9 octavas. El programa puede controlar la envolvente, la afinación y la forma de onda de cada voz,

convirtiendo al COMMODORE 64 en el mejor simulador de instrumentos.

3. Conectable directamente a toda una gama de periféricos, incluyendo unidad de discos, impresora de matriz de puntos o de margarita, plotter, comunicaciones locales y remotas..., y mucho más.

4. Pantalla de alta resolución en color con 320 × 200 puntos directamente direccionables. Capacidad en modo carácter de 25 líneas por 40 columnas.

5. El chip de video, único en su género, permite el uso de 8 «Sprites» (figuras móviles en alta resolución y color). Los «Sprites» pueden moverse independientemente por programa de «pixel» en «pixel».

6. A cada «Sprite» se le asigna por programa un nivel de prioridad en caso de cruce con otro, consiguiendo efectos tridimensionales, existiendo también detección automática de colisiones.

7. Teclado profesional con mayúsculas y minúsculas, más 62 caracteres gráficos, todos ellos disponibles en el teclado y visualizables en 16 colores, en forma normal o bien en video invertido.

8. Encontrará a su disposición una completa gama de programas profesionales, incluyendo proceso de textos, sistemas de información, modelos financieros, contabilidad y muchas más aplicaciones.

9. Están en fase de desarrollo asimismo otros lenguajes tales como LOGO, UCSD PASCAL, COMAL, ASSEMBLER, etc. Todos los programas existentes de la gama COMMODORE, desde el VIC-20 hasta los modelos CBM pueden ser adaptados fácilmente.

10. Posibilidad de inserción de cartuchos con programas grabados en ROM, tanto profesionales como para educación y ocio.

11. Opción de un segundo procesador Z-80 para trabajar con sistema operativo CP/M (R).

EL COMMODORE 64 Y SU MAS DIRECTO COMPETIDOR

OPCION DE BASE	COMMODORE 64	Más directo competidor
Precio	110.000,— ptas.	El doble
Memoria usuario	64 K	48 K
Teclado profesional	SI	SI
Teclado con caracteres gráficos	SI	NO
Minúsculas	SI	NO
Teclas de función	SI	NO
Máxima capacidad disco	170 K a 1 M	143 K
AUDIO		
Generador de sonido	SI	SI
Sintetizador de música	SI	NO
Salida HI-FI	SI	NO
VIDEO		
Salida monitor	SI	SI
Salida para TV	SI	EXTRA
PERIFERICOS		
Cassette	SI	SI
Periféricos inteligentes	SI	SI
Bus serie	SI	NO
SOFTWARE		
Opción CP/M (R)	SI	SI
Ranura cartucho externo	SI	NO

commodore
COMPUTER

PARA MAS INFORMACION
DEL COMMODORE 64,
LLAMAR O ESCRIBIR A:
MICROELECTRONICA Y CONTROL
c/ Taquígrafo Serra, 7, 5º. Barcelona-29
Tel. (93) 250 51 03
c/ Princesa, 47, 3º, G. Madrid-8
Tel. (91) 248 95 70

Nombre ....
Dirección ....
Tel. ....
Población ....

SLOGAN

PROGRAMAS 

Programas Aplicaciones

C-130 Caja 2.000
Dentro de una cuenta bancaria permite abrir hasta 100 subcuentas con su correspondiente debe y haber. Preparado para funcionar si se desea con impresora, unidad de discos y evidentemente cassette. Necesita ampliación 16K. Presentado en caja.

C-136 Dieta 2.000
En función de la edad, sexo, altura, configuración y actividad desempeñada por la persona, da el peso ideal. Pregunta si se desea seguir un régimen, y en función de la actividad física desarrollada da el peso que se debe tener y la pérdida diaria que se debe alcanzar hasta llegar a dicho peso óptimo. Está en castellano y necesita ampliación de 8K. Presentado en caja.

C-139 Vicalc 1.500
El VIC-20 se comporta como una calculadora y utiliza la notación polaca inversa. Manual de instrucciones en castellano.

C-142A Interface de RTTY y CW y cassette de CW . 25.000
Especial para radioaficionados.

C-142B Cassette de RTTY 2.500
Para utilizarlo es necesario el interface de RTTY y CW. Especial radioaficionados.

C-144 Quiz Master 2.000
Programa para preparación de cuestionarios. Especialmente adecuado para escuelas. En castellano. Necesita ampliación de 8K.

Programas de Juegos

C-201 Codemaker 1.500
El VIC-20 inventa un código secreto, teniendo que descubrirlo. Instrucciones en castellano y el programa en inglés. No necesita ampliación.

C-202 Wall Street 1.500
Partiendo de una situación financiera inicial, puede comprar y vender acciones diariamente en función de los precios que van variando. Instrucciones en castellano. Programa en inglés. No necesita ampliación.

C-203 Simple Simon 1.500
El VIC-20 va generando una serie de notas musicales que el usuario debe repetir en la misma secuencia. El grado de dificultad va aumentando y cuando se comete un error se vuelve a iniciar el juego. Instrucciones en castellano. Programas en inglés. No necesita ampliación.

C-2041 Damas 1.500
Juego de Damas especialmente adecuado para niños entre los 7 y los 10 años. En inglés. No necesita ampliación.

C-205 Alien Blitz 1.500
Fabuloso juego de invasores para el VIC-20. No necesita ampliación.

C-206 Kosmic Kamikaze 1.500
Ataque masivo de naves extraterrestres, pudiéndole salvar tan sólo sus habilidades y reflejos. Necesita ampliación de 3K u 8K, dado que hay dos versiones del juego.

C-207 Star Wars 1.500
Tan solo dispone de 1 minuto para destruir al enemigo!!, siendo difícil el tener colocado los invasores en el centro de la pantalla para abatirlos. No necesita ampliación.

C-208 Amok 1.500
Solamente con su habilidad y reflejos, el hombre dirigido por usted podrá esquivar el ataque del enemigo y aniquilarlo. No necesita ampliación.

C-202 The Alien 1.500
¡El comecocos en acción! Necesita una ampliación de 3K.

C-210 Invader Fall 1.500
Defenderá su posición frente al avance progresivo de naves espaciales. Debe conseguir la destrucción total antes de la llegada a sus líneas. No necesita ampliación.

C-211 A-MAZ-ING 1.500
Piérdase por un laberinto poblado de fantasmas. Necesita ampliación de 3K.

C-212 Math-Hurdler, Monster Maze 1.500
Math-Hurdler: Tiene que resolver la operación matemática por usted escogida con el grado de dificultad y rapidez que desee. Monster Maze: Debe llegar al otro extremo del laberinto evitando que sus enemigos se lo coman. No necesita ampliación.

C-213 Golf 1.500
Juegue al golf sentado cómodamente en su casa frente a la pantalla del televisor. Necesita 3K.

C-215 VIC Games II 1.500
Fire: Apague un incendio con un helicóptero.
Draw: Dibuje en la pantalla con alta resolución.
Race: Intente imitar a Fangio en espectacular carrera de coches. No necesita ampliación.

BIBLIOGRAFÍA

Manual usuario VIC-20 800
Nociones de funcionamiento del VIC-20 y del lenguaje BASIC.

Curso Introducción BASIC: Parte I 2.500
De una manera clara y sencilla, partiendo de cero, se va enseñando el lenguaje del futuro. Este libro va acompañado de dos cassettes con 17 programas en castellano.

Curso Introducción BASIC: Parte II 2.500
Profundización de lo tratado en Parte I, especialmente en creación de ficheros, manejo de subrutinas, efectos musicales, aspectos animados, etc.

VIC Revealed 1.500
Libro que explica con detalle el funcionamiento interno del VIC-20. Incluye esquemas. Libro en inglés.

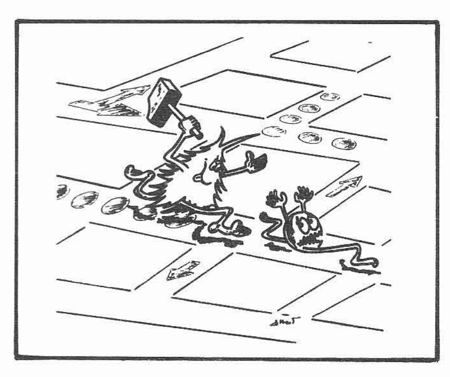
Programers Reference Guide 2.000
Profundización en el manejo del VIC-20. Incluye referencias a rutinas lenguaje máquina. Libro fotocopiado en inglés.

PROGRAMAS COMERCIALIZADOS POR "MICROELECTRÓNICA Y CONTROL, S.A."
Puede encontrarlos en su distribuidor más próximo

microelectrónica
y control s.a



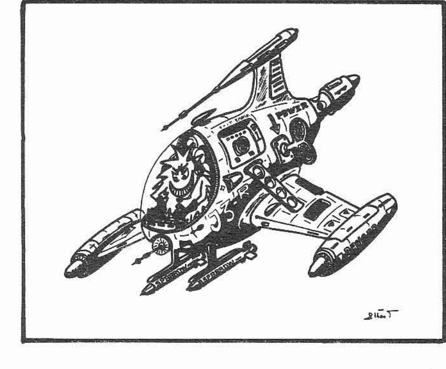




JUEGOS

Pincel para el VIC

por **P. MASATS**



Este programa de trazado gráfico en Alta Resolución está preparado para funcionar en un VIC en su configuración mínima de 5K con 1K de reserva. Cuando ejecutemos el programa, la pantalla se volverá negra (POKE 36879, 8). Deberá esperar unos segundos para que el VIC ejecute los bucles. Pulse la tecla 'D', y un pixel aparecerá en la pantalla; pulsando 'E' desaparecerá.

El pincel puede moverse continuamente o de pixel en pixel. El movimiento continuo está controlado por las teclas de función, mientras que el trazado pixel a pixel está controlado por las teclas ',', '.', '<', '>', y los controles de cursor. El pincel dibujará a la izquierda, a la derecha, hacia arriba, o hacia abajo, y también en las cuatro direcciones en diagonal. La tecla '$' para todo movimiento de pincel. Es posible también mover el pincel sin dibujar pulsando la tecla 'M'. ■

```
READY.

50 POKE56,24:POKE52,24
60 POKE36869,255
70 FORI=7168TO7679:POKEI,0:NEXT
80 POKE36879,8 :PRINTCHR$(147)
90 FORI=7680TO8191:POKEI,160:NEXTI
100 FORL=0TO7:FORM=0TO7
110 POKE7841+M*22+L,L*8+M
120 NEXT:NEXT
130 GETB$
140 IFB$="D"THENC$="D"
150 IFB$="E"THENC$="E"
160 IFB$="M"THENC$="M"
170 IFB$="[HOME ]"THENX=0:Y=0
180 IFB$="S"THENA$=""
190 IFB$="[  F1 ]"ORB$="[  F2 ]"ORB
$="[  F3 ]"ORB$="[  F4 ]"ORB$="[  F5 ]
"ORB$="[  F6 ]"ORB$="[  F7 ]"ORB$="
[  F8 ]"THENA$=B$
200 IFB$="[ CLR ]"THEN220
210 GOTO230
220 FORI=7168TO7679:POKEI,0:NEXT
230 IFB$="[CRSRR]"THENX=X+1
240 IFB$="[CRSRL]"THENX=X-1
250 IFB$="[CRSRU]"THENY=Y-1
260 IFB$="[CRSRD]"THENY=Y+1
270 IFB$=","THENX=X-1:Y=Y+1
280 IFB$="<"THENX=X-1:Y=Y-1
290 IFB$="."THENX=X+1:Y=Y+1
300 IFB$=">"THENX=X+1:Y=Y-1
310 IFA$="[  F1 ]"THENX=X+1
320 IFA$="[  F3 ]"THENY=Y+1
330 IFA$="[  F5 ]"THENX=X-1
340 IFA$="[  F7 ]"THENY=Y-1
350 IFA$="[  F2 ]"THENX=X+1:Y=Y+1
360 IFA$="[  F4 ]"THENX=X-1:Y=Y+1
370 IFA$="[  F6 ]"THENX=X-1:Y=Y-1
380 IFA$="[  F8 ]"THENX=X+1:Y=Y-1
390 IFX<0THENX=0
400 IFX>62THENX=62
410 IFY<0THENY=0
420 IFY>62THENY=62
430 CH=INT(X/8)*8+INT(Y/8)
440 RO=(Y/8-INT(Y/8))*8
450 BY=7168+8*CH+RO
460 BI=7-(X-(INT(X/8)*8))
470 IFC$="D"THENPOKEBY,PEEK(BY)OR(2
↑BI)
480 IFC$="M"THENPOKEBY,0ORPEEK(BY)
490 IFC$="E"THENPOKEBY,0
500 GOTO130
READY.
```

Ahora el VIC-20 y CBM 64 pueden comunicarse con Periféricos Commodore

USUARIOS DEL VIC-20 y CBM 64
¿Le gustaría tener acceso a cualquiera de los siguientes perifericos desde su computador?

* Discos de 1/3 megabyte (Commodore 4040)
* Discos de 1 megabyte (Commodore 8050)
* Discos de 2 megabyte (Commodore 8250)
* Discos de 10 megabyte (Commodore 9090 discos duros)
* Impresoras con IEEE y RS 232 matricial y margarita
* Instrumentos IEEE, como voltimetros, plotters, etc.

Ahora ya no se queda Vd. limitado por el VIC y la serie de los 64. Simplemente añadiendo un INTERPOD puede Vd. aumentar ampliamente la potencia de su VIC-20 y usándolo con el nuevo CBM 64, el INTERPOD convertira su computador en un sistema realmente potente.

Con el INTERPOD, el VIC-20 y el CBM 64 son capaces de llevar a cabo un software de calidad y profesional, tales como proceso de datos, Contabilidad, Control de stok y mucho más ...
INTERPOD está capacitado para trabajar con cualquier software. No se necesitan ningún comando extra y no afecta bajo ningún aspecto para nada a su computador.
Usar el INTERPOD es tan simple y fácil como:

* Enchufar el INTERPOD en la salida de serie de su computador, pongalo en funcionamiento y ya está Vd. listo para comunicarse con cualquier periferico de la serie IEEE y cualquier impresora RS232.

ESTO ES EL INTERPOD.

Importador para España:

C/. BALMES. 13
Tel. (971) 24 54 04
Palma



Es un producto de Oxford Computer Systems (Software) Ltd. U.K.

JUEGOS

Laberinto para PET-CBM 4000 y 8000

por **R. PARDO**

El objetivo de este juego es salir del laberinto en el menor tiempo posible. Se añade realismo al juego mostrando sólo siete posiciones en alguna de las cuatro posibles direcciones de movimiento. Es como si dispusiera de una linterna en una noche oscura, que le muestra varias vías de salida y con el desplazamiento éstas se van desvaneciendo. El movimiento se genera pulsando una tecla para movernos en una dirección determinada.

Para movernos pulsaremos las siguientes teclas:

'8'= movimiento hacia arriba
'2'= movimiento hacia abajo
'4'= movimiento hacia la izquierda
'6'= movimiento hacia la derecha

Tenemos también la posibilidad de qúe el ordenador nos muestre la totalidad del laberinto, pero esto nos costará puntos. Para mostrar el laberinto deberá pulsar '?'.

Este juego necesita aproximadamente unos 45 segundos para generar el laberinto.

Para adaptar este juego a un ordenador de la serie CBM 8000 deberá cambiar en la línea 605 el valor de VL de 40 a 80. ■

```
READY.

10 GOTO400

100 REM LABERINTO PARA PET 4000 & 8
000

110 GOSUB730:PRINTCT:FORJ=0TO3:D=A:
C=S2

120 FORI=1TO7:POKEC,M(ABS(D))

130 POKEC-E(J),M(ABS(D-D(J))):POKEC
+E(J),M(D+D(J))

140 IFM(D)=WLTHEN160

150 D=D+A(J)/2:C=C+E(J+1):NEXTI

160 NEXTJ:POKES2,240:M(A)=240:RETUR
N

200 REM

210 FORI=1TOH:FORJ=2TOH+1:POKES2,J

220 M((I*(H+2))+J)=WL:NEXTJ:NEXTI

300 REM GENERA LABERINTO

310 M(A)=4

320 J=INT(RND(R)*4):Z=J:POKES2,J

330 B=ABS(A+ABS(A(ABS(J)))):IFM(B)=
WLTHENM(B)=J:M(A+A(J)/2)=HL:A=B:GOT
O320

340 J=(J+1)*-(J<3):IFJ<>ZTHEN330

350 J=M(A):M(A)=HL:IFJ<4THENA=A-A(J
):GOTO320

360 T1=(3*H)+5:T2=INT(RND(R)*2):IFI
NT(T2/2)*2=T2THENT1=(2*H)+6

362 Q1=-1:Q2=-(H+1):Q3=H+3

364 IFT1=(2*H)+6THENQ1=-Q1:Q2=-Q2:Q
3=-Q3

366 Z=INT(RND(R)*(H-3))*(H+2)+T1:IF
M(Z+Q1)<>HLTHEN366

370 M(Z)=69:M(Z+Q2)=WL:M(Z-Q1)=WL:M
(Z+Q3)=WL

380 FORI=1TOH

382 M(3*(H+2)+4+INT(RND(R)*(H-5))*(
H+2)+INT(RND(R)*(H-5)))=HL

384 NEXTI:RETURN

400 REM

410 GOSUB600:GOSUB900:GOSUB730:GOSU
B200:GOSUB100

415 GETA$:IFA$=""THENCT=CT+1:GOTO41
5

420 J=-((A$="8")+2*(A$="4")+3*(A$="
2"))

425 IFA$="?"THEN500

435 A2=A+A(J)/2

440 IFM(A2)=69THEN800

445 IFM(A2)<>HLTHEN415

450 M(A)=HL:A=A2:GOSUB100:GOTO415

500 REM

510 GOSUB730:F=0:FORI=1TOH:FORJ=2TO
H+1:L=(I*(H+2))+J

520 POKES+J+F,M(L):NEXTJ

530 F=F+FF:NEXTI

540 FORI=1TO200:NEXTI:CT=CT+500:GOS
UB100:GOTO415

600 REM

605 VL=40:H=23:FF=VL:REM PARA 80 CO
LUMNAS CAMBIE EL VALOR VL POR 80

610 A(0)=2:A(1)=-(H+2)*2:A(2)=-2:A(
3)=(H+2*2)*2

620 D(0)=H+2:D(1)=1:D(2)=-(H+2):D(3
)=-1

630 E(0)=VL:E(1)=1:E(2)=-VL:E(3)=-1
:E(4)=VL

640 WL=160:HL=32:S2=32768+VL*12+INT
(VL/2):CT=0

650 S=S2-VL*((H+1)/2)+FF-(H+3)/2

660 A=(H+2)*(H+1)/2+(H+3)/2:DIMM(70
0):DIMB(700)

730 PRINT"[ CLR ]";:RETURN

800 REM

810 GOSUB730:PRINT"SU PUNTUACION ";
CT:END

900 REM

910 REM

920 PRINT"  LABERINTO":PRINT

930 PRINT"SALDRA RAPIDAMENTE   ":PR
INT"USANDO LAS   ":PRINT"TECLAS DE
CONTROL."

940 PRINT:PRINT"'8' ARRIBA,"

950 PRINT"'2' ABAJO,"

960 PRINT"'4' IZQUIERDA,":PRINT"'6'
 DERECHA,":PRINT

970 PRINT"'?' MUESTRA EL LABERINTO"
:PRINT"PERO CUESTA PUNTOS.":PRINT

980 R=0:PRINT"PULSE [RVSON]RETURN[RVSOF]
"

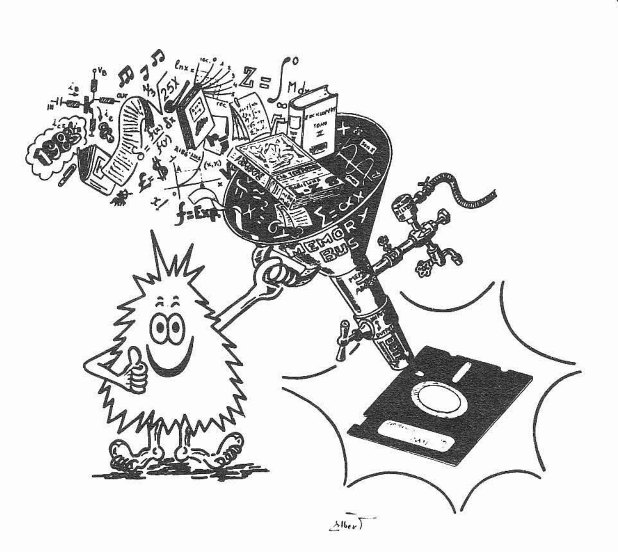
985 GETA$:IFA$=""THEN985

990 RETURN

READY.
```

almacenamiento en discos (I)

por **MANUEL AMADO**
(M.E.C. SOFT)



INTRODUCCIÓN

El presente artículo es el primero de una serie de introducción a las técnicas de manejo y tratamiento de datos en las unidades de disco COMMODORE, siendo válidas igualmente para los monofloppys 1540 y 1541 de la serie de ordenadores personales. Pero sobre todo, esta serie de artículos estará enfocada principalmente al tratamiento de los dual floppy COMMODORE (4040, 8050, 8250).

Empecemos considerando dos conceptos de la palabra fichero que se usan conjuntamente en el desarrollo de una aplicación informática y que corresponden —en el caso de trabajar con floppys COMMODORE— a dos realidades diferenciadas.

Fichero Lógico.—Es un tipo de organización de los datos almacenados. Está formado por una serie de elementos llamados registros, y éstos, a su vez, se dividen en campos, que acostumbran a contener un mismo tipo de dato cada uno. Se corresponde totalmente con el concepto de fichero desde el punto de vista de aplicaciones de gestión. Por ejemplo, si tenemos un fichero llamado CLIENTES, éste puede constar de 500 registros, en donde cada registro tendrá un Nombre, Dirección y Teléfono, o sea 3 Campos diferentes.

Fichero Disco o **Fichero Físico.**—En este caso nos referimos a un tipo determinado de almacenamiento FÍSICO de la información en un soporte magnético. Se trata de un floppy. El programador será el que organizará convenientemente el fichero en los diferentes registros y campos de que conste su fichero lógico. COMMODORE tiene tipos distintos de ficheros de disco, que son:

FICHERO SECUENCIAL

En él los datos están almacenados uno a continuación de otro, accediéndose sucesivamente a los diferentes datos almacenados. Para leer un dato hay que empezar a leer desde el principio del fichero hasta encontrar el que se busca.

FICHERO PROGRAMA

Este tipo de fichero se utiliza para almacenar los programas en el disco. Su característica más sobresaliente es que se puede cargar y grabar de una sola vez desde memoria, mediante las instrucciones LOAD y SAVE.

FICHERO RANDOM
o de ACCESO DIRECTO

Es también un fichero de datos y su principal característica es que se trata de un fichero diseñado por el propio programador, y en el que se accede directamente a un determinado bloque del disco, indicando en cada caso la pista y sector deseados.

FICHERO RELATIVO

Igualmente es un fichero de datos y el método normal de acceso a los datos es el acceso a un NÚMERO DE REGISTRO determinado, que es en el que se halla el dato que deseamos leer.

En algunos de ellos, relativos o random, el fichero sí que está dividido en registros o fichas, y son accesibles individualmente. Pero en otros casos no, como el secuencial, en el que es el programador el que tiene que establecer la organización lógica del fichero en sus diferentes registros y campos.

FICHEROS SECUENCIALES

Es el tipo de fichero disco más elemental, almacenándose los datos uno detrás de otro, SECUENCIALMENTE según el orden en que se hayan grabado.

Su principal característica estriba en que para leer un determinado dato, hay que leer todo el fichero desde el principio hasta que se encuentra el dato deseado. Asimismo, si se desean añadir nuevos datos, habrá que releer y grabar totalmente el contenido anterior del fichero antes de proceder a la inserción del nuevo dato, y grabar a continuación todo el fichero otra vez. Es, en realidad, como un fichero de datos de una cinta cassette pero con la ventaja de un acceso muchísimo más rápido y que cada fichero secuencial sí que puede accederse directamente, sin tener que leer todo el disco.

1.1. COMANDOS BASIC DE ACCESO Y TRATAMIENTO DE LOS FICHEROS SECUENCIALES

Naturalmente, la anterior disertación teórica es realmente bonita, pero ¿cómo se almacena y cómo se puede gestionar la información guardada en un fichero secuencial?

Esto es lo que vamos a ver en los siguientes apartados, empezando por los comandos BASIC que van a permitir manejar dichos ficheros. Pero antes de pasar a describirlos, voy a des-

almacenamiento en discos

(viene de la pág. anterior)

arrollar una corta explicación de la filosofía de trabajo de los ficheros propios de COMMODORE. Me explico:

En la gestión de todo fichero, se distinguen cuatro operaciones totalmente diferenciadas y que pueden realizarse o no independientemente unas de otras (según el tipo de fichero, de aplicación, etc.):

A) CREACION DEL FICHERO
B) GRABACION DE DATOS
C) LECTURA O ACCESO A LOS DATOS GRABADOS
D) MODIFICACION Y/O BORRADO DE LOS DATOS.

A) CREACIÓN DEL FICHERO

Estamos aquí ante un concepto, el de CREACIÓN de un fichero, que seguramente ya será familiar para muchos de vosotros. Pues bien, y en el caso de ficheros CBM, la creación de un fichero significa simplemente la creación de la ENTRADA DE DIRECTORIO correspondiente a dicho fichero en el diskette, y la reserva del PRIMER bloque de datos en el BAM. El significado y estructura del directorio y BAM del disco se verán en profundidad en un artículo próximo; pero, en pocas palabras, podemos decir que el directorio de un disco es un bloque de control en el cual se almacenan los nombres de los ficheros que se hallan en cada momento en el disco en cuestión y la dirección del disco en donde se halla dicho fichero. El BAM es un bloque especial de control, que sirve para que el DOS (Disk Operating System o Sistema Operativo del Disco) sepa si un bloque de disco está ocupado (asignado a algún fichero) o libre.

Pues bien, cuando se desea almacenar información en el disco, lo primero que hay que hacer es crear el fichero correspondiente, para que luego el DOS sepa en dónde ha de ir a buscar los datos que nosotros deseemos tratar.

Todo esto se consigue mediante un simple comando BASIC, el comando OPEN. Para la creación de un fichero secuencial, su formato es:

OPEN fichero, periférico, canal, «drive:nombre, s(eq), w(rite)»

Significado de cada parámetro:

— **Fichero:** número de fichero lógico asignado al fichero que se desea crear y que servirá de referencia para que los comandos PRINT #, GET # e INPUT #, sepan cuál es el que se desea procesar.

— **Periférico:** número de periférico de la unidad de disco.

— **Canal:** número LÓGICO de canal del floppy a través del cual se van a transmitir los datos desde el soporte magnético. El número físico del buffer del floppy asociado a dicho canal lo determina el DOS.

— **Drive:** número de drive en donde se halla el fichero (0 ó 1).

— **S(EQ):** indica que el tipo de fichero que se va a abrir es S(EQ)uencial. Se puede escribir «seq» o bien abreviadamente «s».

— **W(rite):** indica que se van a enviar datos desde la CPU al fichero. Se puede escribir «write» o abreviadamente «w». La razón de que para crear un fichero secuencial se tenga que realizar en MODO escritura, estriba en que, si se realiza para lectura, el fichero tiene que existir y, por lo tanto, nos daría un error de no existencia de fichero.

B) GRABACIÓN DE DATOS

Una vez abierto el fichero en modo escritura, se puede proceder a la grabación de los datos que deseemos almacenar.

La grabación de datos se realiza mediante el comando:

PRINT # NF, (datos)

Donde NF es el número de fichero lógico que se había asignado en el momento de la apertura (creación) del fichero que queremos utilizar, y (datos) los datos que se desean grabar, ya sea en forma de variables (numéricas o alfanuméricas), o literales especificados en la misma instrucción. O sea:

PRINT # 2, a$, o bien:
PRINT # 2, «pepe».

En un caso estamos escribiendo en el fichero número 2 el contenido de a$, y en el otro caso simplemente estamos grabando en el fichero el literal «pepe».

Pues bien, ya sabemos cómo se escriben en el disco los datos que queramos, pero..., ¿bajo qué formato se graban en el disco, y cuáles son las diferentes posibilidades de formación de los datos escritos con el comando PRINT #? Pues bien, en los próximos párrafos voy a intentar aclarar estos puntos.

SEPARADORES DEL COMANDO PRINT # NF,

Supongamos que ejecutamos la siguiente sentencia:

A$="PEPE":B$="CANAS":PRINT # 3, A$:PRINT # 3,B$

El contenido en disco del fichero será el de la Tabla 1.

Donde cr significa el carácter de código 13, CHR$(13) o CARRIAGE RETURN, que se graba al final de la ejecución de cada comando PRINT # NF, siempre y cuando este comando no termine en una coma o un punto y coma (, o ;). Este retorno de carro es el que separa y establece la diferencia

TABLA 1

CHAR	P	E	P	E	cr	C	A	N	A	S	cr	eof
Byte	1	2	3	4	5	6	7	8	9	10	11	12

entre un dato y otro y va a ser utilizado por el comando de lectura, INPUT # NF, para saber dónde acaba el dato que se desea leer en cada momento. eof es la marca de FIN DE FICHERO o END OF FILE, y que nos indicará cuándo hemos llegado al final del fichero, provocando que la variable de estado ST pase a valer 64.

Ahora bien, es realmente engorroso el tener que ejecutar un comando PRINT # NF para cada dato que deseemos grabar. Pues bien, nada más sencillo que poder grabar diferentes datos en un mismo comando PRINT # NF. Veamos qué sucede al ejecutar la sentencia:

A$ = "PEPE": B$ = "CASA": C$ = "SOL": PRINT # 3,A$;B$; CHR$(13);C$

El contenido en el disco será el de la Tabla 2.

Observamos que A$ (PEPE) y B$ (CASA), nos las ha grabado juntas, sin ninguna separación que nos permita IDENTIFICAR dónde acaba el dato A$ y empieza B$. Por esta causa al leer el dato mediante el comando INPUT # NF (no desesperar, que hablaremos de él más adelante con detalle), nos leerá A$ y B$ como si fuese un solo dato, o sea, PEPECASA. Todo ello se debe a que al grabar varios datos separados entre sí por punto y coma, ";", éstos se graban uno detrás de otro sin separador ni separación de ninguna clase. Pero podemos introducir el separador, CHR$(13), ejecutando precisamente un PRINT # 3,CHR$(13);, tal y como hemos hecho al grabar C$, que ha quedado separada de B$ por el cr que hemos grabado con CHR$(13).

TABLA 2

CHAR	P	E	P	E	C	A	S	A	cr	S	O	L	cr	eof
Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14

TABLA 3

CHAR	S	I										N	O	cr	eof
Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Pues bien, ésta es la forma para grabar una lista de datos con un solo comando PRINT # NF, y de forma que estos datos queden diferenciados para poder ser leídos posteriormente por separado.

Veamos ahora qué efecto produce el utilizar como separador una coma en lugar del punto y coma. Si ejecutamos:

A$="SI": B$="NO":PRINT # 3,A$,B$

el resultado es el de la Tabla 3.

Se observa que al utilizar como separador la coma, nos graba en disco 10 espacios (chr$(32)), entre cada variable separada por la coma (para el caso del 8050, y, según me acaba de confirmar la Redacción, 11 espacios en el caso del 1541). Consecuentemente, es realmente desaconsejable el uso de la coma como separador. Finalmente, hay que indicar que nos podemos ahorrar el uso del punto y coma como separador en el caso de la grabación de variables ALFANUMÉRICAS y es preferible el poner en forma de variable alfanumérica el CHR$(13).

Por ejemplo:

RT$=CHR$(13):A$="PEPE":B$="SOL":C=12:PRINT# 3, ARTBRT;C

TABLA 4

CHAR	P	E	P	E	cr	S	O	L	cr		1	2	cr	eof
Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14

TABLA 5

CHAR	P	E	P	E		1	2	3	4	5	6		cr	eof
Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Quedando en disco como se ve en la Tabla 4.

CODIFICACIÓN DE LOS DATOS GRABADOS

Los datos que se van grabando en el fichero mediante la instrucción PRINT # NF, se graban en el correspondiente código ASCII-CBM, de la misma forma en que están almacenados los strings en memoria. En el caso de la escritura de variables numéricas, se transforma su representación en coma flotante de cinco bytes en memoria a su correspondiente representación en forma de string ascii (tal y como aparecen en pantalla al realizar un print m1, donde m1 es una variable numérica), de forma parecida a como si el comando PRINT # NF grabase en disco en vez de m1 (tal y como se guarda en memoria), str$(m1). Pero además, el comando PRINT # NF escribe una marca en el disco para que, a la hora de realizar una lectura mediante el comando INPUT #, éste pueda diferenciar si el dato que está leyendo es *numérico o alfanumérico*. Para ello, además de escribir delante del número un carácter space ,chr$(32)) si el número no tiene signo, graba al final del número un carácter space (chr$(32)). Por ejemplo, si ejecutamos la sentencia:

A$="PEPE":B=123456:PRINT#3,A$;B

Tenemos cuanto se indica en la Tabla 5.

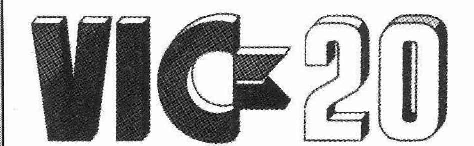
Donde el espacio en en realidad un CHR$(32). ■

TABLA DE EQUIVALENCIAS DEC-HEX-ASCII-PANTALLA-MNEMÓNICOS 6502 (VIC-20) II

DECIMAL	HEX	ASCII	PANTALLA	BASIC	6502	DECIMAL
128	80		r-O	END		128
129	81		r-A	FOR	STA(I,X)	129
130	82		r-B	NEXT		130
131	83		r-C	DATA		131
132	84		r-D	INPUT #	STY Z	132
133	85		r-E	INPUT	STA Z	133
134	86		r-F	DIM	STX Z	134
135	87		r-G	READ		135
136	88		r-H	LET	DEY	136
137	89		r-I	GOTO		137
138	8A		r-J	RUN	TXA	138
139	8B		r-K	IF		139
140	8C		r-L	RESTORE	STY	140
141	8D	car ret	r-M	GOSUB	STA	141
142	8E		r-N	RETURN	STX	142
143	8F		r-O	REM		143
144	90		r-P	STOP	BCC	144
145	91	cur up	r-Q	ON	STA(I),Y	145
146	92	rvs off	r-R	WAIT		146
147	93	clear	r-S	LOAD		147
148	94	insert	r-T	SAVE	STY Z,X	148
149	95		r-U	VERIFY	STA Z,X	149
150	96		r-V	DEF	STX Z,Y	150
151	97		r-W	POKE		151
152	98		r-X	PRINT #	TYA	152
153	99		r-Y	PRINT	STA Y	153
154	9A		r-Z	CONT	TXS	154
155	9B		r-[	LIST		155
156	9C		r-\	CLR		156
157	9D	cur left	r-]	CMD	STA X	157
158	9E		r-↑	SYS		158
159	9F		r-←	OPEN		159
160	A0			CLOSE	LDY #	160
161	A1		r-!	GET	LDA(I,X)	161
162	A2		r-"	NEW	LDX #	162
163	A3		r-#	TAB(		163
164	A4		r-$	TO	LDY Z	164
165	A5		r-%	FN	LDA Z	165
166	A6		r-&	SPC(	LDX Z	166
167	A7		r-'	THEN		167
168	A8		r-(	NOT	TAY	168
169	A9	,	r-)	STEP	LDA #	169
170	AA		r-*	+	TAX	170
171	AB		r-+	-		171
172	AC		r-,	*	LDY	172
173	AD		r--	/	LDA	173
174	AE		r-.		LDX	174
175	AF		r-/	AND		175
176	B0		r-Ø	OR	BCS	176
177	B1		r-1		LDA(I),Y	177
178	B2		r-2	=		178
179	B3		r-3			179
180	B4		r-4	SGN	LDY Z,X	180
181	B5		r-5	INT	LDA Z,X	181
182	B6		r-6	ABS	LDX Z,Y	182
183	B7		r-7	USR		183
184	B8		r-8	FRE	CLV	184
185	B9		r-9	POS	LDA Y	185
186	BA	,	r-:	SQR	TSX	186
187	BB		r-;	RND		187
188	BC		r-	LOG	LDY X	188
189	BD		r-=	EXP	LDA X	189
190	BE		r-	COS	LDX Y	190
191	BF		r-?	SIN		191

DECIMAL	HEX	ASCII	PANTALLA	BASIC	6502	DECIMAL
192	C0			TAN	CPY #	192
193	C1	, a		ATN	CMP(I),X	193
194	C2	, b		PEEK		194
195	C3	, c		LEN		195
196	C4	, d		STR$	CPY Z	196
197	C5	, e		VAL	CMP Z	197
198	C6	, f		ASC	DEC Z	198
199	C7	, g		CHR$		199
200	C8	, h		LEFT$	INY	200
201	C9	, i		RIGHT$	CMP #	201
202	CA	, j		MID$	DEX	202
203	CB	, k				203
204	CC	, l			CYP	204
205	CD	, m			CMP	205
206	CE	, n			DEC	206
207	CF	, o				207
208	D0	, p			BNE	208
209	D1	, q			CMP(I),Y	209
210	D2	, r				210
211	D3	, s				211
212	D4	, t	Reverse of ASCII			212
213	D5	, u			CMP Z,X	213
214	D6	, v			DEC Z,X	214
215	D7	, w				215
216	D8	, x			CLD	216
217	D9	, y			CMP Y	217
218	DA	, z				218
219	DB					219
220	DC					220
221	DD				CMP X	221
222	DE	,			DEC X	222
223	DF	,				223
224	E0				CPX #	224
225	E1				SBC(I),X	225
226	E2					226
227	E3					227
228	E4				CPX Z	228
229	E5				SBC Z	229
230	E6				INC Z	230
231	E7					231
232	E8				INX	232
233	E9		,		SBC #	233
234	EA				NOP	234
235	EB					235
236	EC				CPX	236
237	ED				SBC	237
238	EE				INC	238
239	EF					239
240	F0				BEQ	240
241	F1				SBC(I),Y	241
242	F2					242
243	F3					243
244	F4					244
245	F5				SBC Z,X	245
246	F6				INC Z,X	246
247	F7					247
248	F8				SED	248
249	F9				SBC Y	249
250	FA		,			250
251	FB					251
252	FC					252
253	FD				SBC X	253
254	FE				INC X	254
255	FF			π		255

Depósito Legal Sep. B. 2133-1958 - Altés, s.l., Caballero 87, Barcelona



Microprocesador: 6502 de MOS TECHNOLOGY de 8 bits.
Memoria: 5 Kbytes de RAM ampliables a 32 K 20 Kbytes de ROM ampliables a 28 K
Pantalla: 23 lineas de 22 caracteres Modulador para conectar a un televisor normal. Salida para monitor de video.
Colores: 8 para el marco, 16 para el fondo de la pantalla y ocho para los caracteres individuales, video inverso.
Gráficos: Semi-gráficos por teclado y alta resolución por redefinición del generador de caracteres (situándolo en RAM). Definición de 176 por 184 puntos.
Teclado: Tipo QWERTY de 62 teclas más cuatro de función definibles por el usuario.
Sonido: Tres voces de tres octavas cada una decaladas una octava entre sí, resultando una extensión total de cinco octavas. Un generador de ruido aleatorio afinable para efectos especiales, un control general de volumen.
Programación: Lenguaje BASIC, intérprete residente en ROM de 8 K. Posibilidad de interceptar las funciones del Basic para crear nuevas instrucciones «a medida». El Basic del Vic es uno de los más rápidos actualmente en el mercado.
Complementos: Port de usuario de 8 bits entrada/salida más dos señales de sincronismo.
Bus de expansión para ampliaciones de memoria y periféricos.
Port de juegos con conexión para dos potenciómetros (paddles), y una palanca de juegos (joystick).
Almacenamiento de masa: Unidad de cassette C2N de diseño especial para registrar programas y datos (ficheros secuenciales).

VIC-1540 UNIDAD DE DISCO

Capacidad total: 174848 bytes por disco.
Secuencial: 168656 bytes por disco.
Entradas de directorio: 144 por disco.
Sectores por pista: De 17 a 21.
Bytes por sector: 256.
Pistas: 35.
Bloques: 683 (644 bloques libres).
Soportes de información: Discos estandar de 5 1/4 pulgadas, de una sola cara y densidad simple.
Sistema operativo: DOS de COMMODORE inteligente (tiene procesador propio y no ocupa memoria del ordenador central).

VIC-1515 IMPRESORA

Método de impresión: Matriz de 5×7 puntos, impacto por un solo martillo.
Modo caracteres: Mayúsculas y minúsculas, símbolos, números y caracteres gráficos del VIC-20.
Modo gráfico: Puntos direccionables (bit image). Siete puntos verticales por columna, 480 columna máximo.
Velocidad: 30 caracteres/segundo, de izquierda a derecha, unidireccional.
Caracteres/Linea: Máximo 80. (Posibilidad de impresión en doble ancho).
Espaciado entre lineas: 6 lineas/pulgada -modo caracteres, 9 lineas/pulgadas - modo gráfico.
Velocidad de salto de lineas: 5 saltos/seg. -modo caracteres, 7,5 saltos/seg. - modo gráfico.
Alimentación de papel: Arrastre por tractor.
Ancho de papel: Entre 4,5 y 8 pulgadas.
Copias: Original más dos copias.

CARTUCHOS

Ayuda programador: Este cartucho facilita la edición y depuración de programas en Basic. Instrucciones y comandos: RENUMBER, MERGE, FIND, CHANGE, DELETE, AUTO, TRACE, STEP, OFF, KEY, EDIT, PROG, DUMP, HELP y KILL.

Super expander: Intercepta el Basic del VIC permitiendo incrementar sus instrucciones y comandos en aplicaciones gráficas, de sonido y juegos. Instrucciones y comandos: KEY, GRAPHIC, COLOR, POINT, REGION, DRAW, CIRCLE, PAINT, CHAR, SCNCLR, SOUND, RGR, RCOLR, RDOT, RPOT, RPEN, RJOY y RSND.

Monitor de lenguaje máquina: Este monitor altamente sofisticado facilita enormemente la depuración de programas en lenguaje máquina, es ideal como complemento del Basic para redactar y poner en marcha rutinas de alta velocidad y manejo de datos en tiempo real. Instrucciones y comandos: ASSEMBLE, BREAKPOINT, DISASSEMBLE, ENABLE VIRTUAL ZERO PAGE, FILL MEMORY, GO, HUNT, INTERPRET, JUMP TO SUBROUTINE, LOAD, MEMORY, NUMBER, QUICK TRACE, REGISTERS, REMOVE BREACPOINTS, SAVE, TRANSFER, WALK y EXIT TO BASIC.

Además existen cartuchos de ampliación de memoria de 3, 8 y 16 Kbytes.

CURSO DE INTRODUCCION AL BASIC PARTE I:

En forma de libro se ha editado la primera parte de un curso de Basic que parte «de cero» y está basado en el VIC-20. Va acompañado de dos cassettes con programas y ejercicios para autocontrol de los progresos en el aprendizaje.

MODULO DE EXPANSION DE MEMORIA:

Acabado en metal de gran robustez, permite la conexión de un máximo de 6 cartuchos simultáneamente, aloja al VIC y al modulador de vídeo y permite colocar encima el televisor, tiene alojamiento para accesorios y asegura una óptima conexión del VIC a sus periféricos.







microelectrónica y control, s. a.
Taquígrafo Serra, 7 5.° Telf. 250 51 03. BARCELONA-29
Princesa, 47 3.° G. Telf. 248 95 70. MADRID-8